Multiplicação de Matrizes em Python

Turma: 3º Ano do Ensino Técnico Integrado

Equipe 2

Objetivos

1. Implementar multiplicação de matrizes em Python usando dois métodos:
   - **Python puro** (sem bibliotecas externas).
   - **Numpy** (biblioteca otimizada para cálculos matriciais).
2. Gerar matrizes aleatórias de dimensões específicas (3x2 e 2x3).

Exemplo Visual: Multiplicação de Matrizes

Matriz A (3x2):

```
[1, 2]
[3, 4]
[5, 6]
```

Matriz B (2x3):

```
[7, 8, 9]
[10, 11, 12]
```

Cálculo de C[0][0]:
(1*7) + (2*10) = 7 + 20 = 27

Resultado Final (3x3):

```
[27, 30, 33]
[61, 68, 75]
[95, 106, 117]
```

Método 1: Python Puro (Matrizes Aleatórias)

```python
import random

# Gerar matriz 3x2 com valores entre 1 e 10
matriz_A = [[random.randint(1, 10) for _ in range(2)] for _ in range(3)]

# Gerar matriz 2x3 com valores entre 1 e 10
matriz_B = [[random.randint(1, 10) for _ in range(3)] for _ in range(2)]

# Função de multiplicação
def multiplicar_matrizes(A, B):
    # Verifica se as matrizes são compatíveis
    if len(A[0]) != len(B):
        return "Erro: Dimensões incompatíveis!"

    # Cria matriz resultado preenchida com zeros
    resultado = [[0 for _ in range(len(B[0]))] for _ in range(len(A))]

    # Loop para calcular cada elemento
    for i_linha in range(len(A)):        # Linhas de A
        for j_coluna in range(len(B[0])):  # Colunas de B
            for k in range(len(B)):        # Produto escalar
                resultado[i_linha][j_coluna] += A[i_linha][k] * B[k][j_coluna]
    return resultado

# Exemplo de uso
print("Matriz A:", matriz_A)
print("Matriz B:", matriz_B)
print("Resultado:", multiplicar_matrizes(matriz_A, matriz_B))
```

Método 2: Numpy (Matrizes Aleatórias)

```python
import numpy as np

# Gerar matriz 3x2
A = np.random.randint(1, 10, size=(3, 2))

# Gerar matriz 2x3
B = np.random.randint(1, 10, size=(2, 3))

# Multiplicação com numpy
resultado = A @ B

print("Matriz A:\n", A)
print("Matriz B:\n", B)
print("Resultado:\n", resultado)
```

Comparação dos Métodos

Característica	Python Puro	Numpy
Velocidade	Lento para matrizes grandes	Rápido
Facilidade de Uso	Requer loops manuais	Sintaxe simplificada
Geração Aleatória	Via list comprehensions	Funções otimizadas

Dicas

- Use `random.seed(42)` para gerar matrizes com valores fixos (útil para testes).
- Em numpy, use `C = np.matmul(A, B)` como alternativa a `@`.

Referências

1. Documentação do Numpy: numpy.org
2. Livro: *Python para Análise de Dados* (Wes McKinney).